The Landscape of Structural Graph Parameters

Michael Lampis
KTH Royal Institute of Technology

November 18th, 2011

Introduction

FPT theory in 30 seconds

- Most problems are NP-hard $\rightarrow$ need exp time in the worst case
- They may be easily solvable in some special cases
  - Typically for graph problems, when the graph is a tree
- What about the almost easy cases?
  - We consider the concept of "distance from triviality"

FPT theory in 30 seconds

- Most problems are NP-hard $\rightarrow$ need exp time in the worst case
- They may be easily solvable in some special cases
  - Typically for graph problems, when the graph is a tree
- What about the almost easy cases?
  - We consider the concept of “distance from triviality”
- Examples:

FPT theory in 30 seconds

- Most problems are NP-hard $\rightarrow$ need exp time in the worst case
- They may be easily solvable in some special cases
  - Typically for graph problems, when the graph is a tree
- What about the almost easy cases?
  - We consider the concept of “distance from triviality”
- Examples:
  - Satisfying $\frac{7}{8}m$ of the clauses of a 3-CNF formula
  - Satisfying $\frac{7}{8}m+k$ of the clauses of a 3-CNF formula

FPT theory in 30 seconds

- Most problems are NP-hard $\rightarrow$ need exp time in the worst case
- They may be easily solvable in some special cases
  - Typically for graph problems, when the graph is a tree
- What about the almost easy cases?
  - We consider the concept of “distance from triviality”
- Examples:
  - Euclidean TSP on a convex set of points
  - Euclidean TSP when all but k of the points lie on the convex hull

FPT theory in 30 seconds

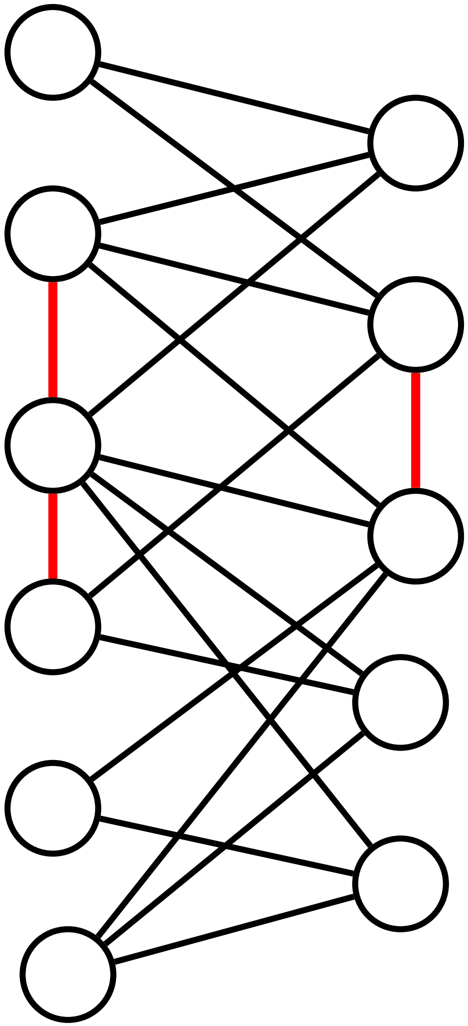
- Most problems are NP-hard $\rightarrow$ need exp time in the worst case
- They may be easily solvable in some special cases
  - Typically for graph problems, when the graph is a tree
- What about the almost easy cases?
  - We consider the concept of “distance from triviality”
- Examples:
  - Vertex Cover on bipartite graphs
  - Vertex Cover on graphs with small bipartization number

Vertex Cover

- Vertex Cover is NP-hard in general.
- It is easy (in P) on bipartite graphs.
  - Maximum matching, König's theorem
- What about almost bipartite graphs?
  - Is there an efficient algorithm for Vertex Cover on graphs where the number of vertices/edges one needs to delete to make the input graph bipartite is small?
- Assume for now that some small bipartizing set is given.

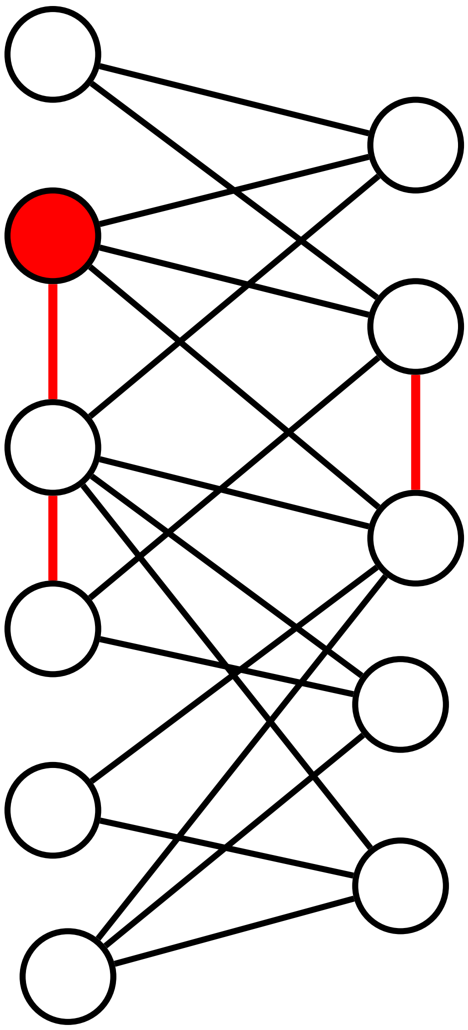
Search tree algorithm

- Suppose we have an almost bipartite graph. We cannot use König's theorem to find its minimum vertex cover.
- However, we can try to get rid of the offending vertices/edges.



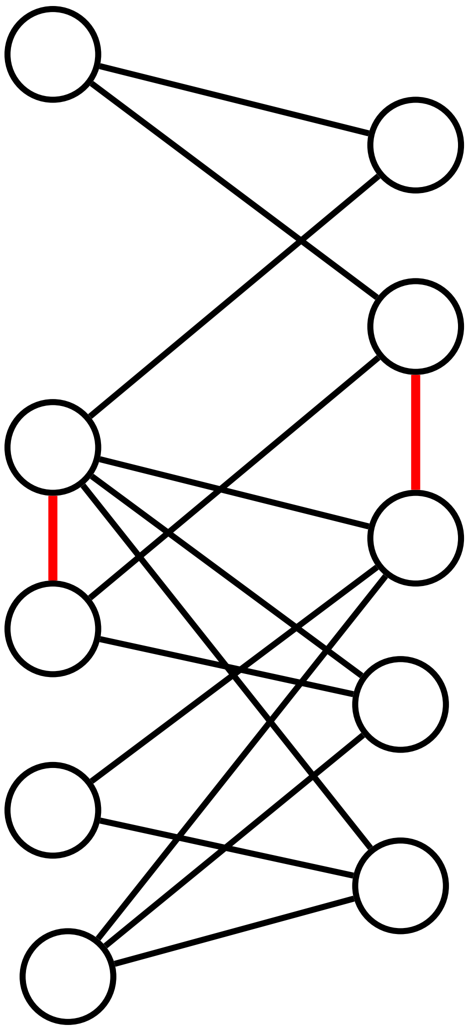
Search tree algorithm

- Pick an offending edge. Either its first endpoint must be in the optimal vertex cover . . .



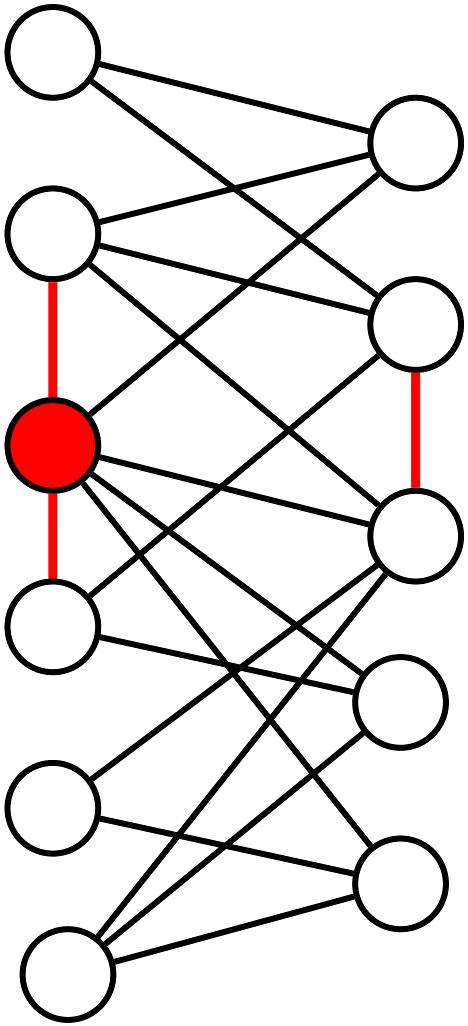
Search tree algorithm

- Pick an offending edge. Either its first endpoint must be in the optimal vertex cover . . .
- So, we should remove it. . .



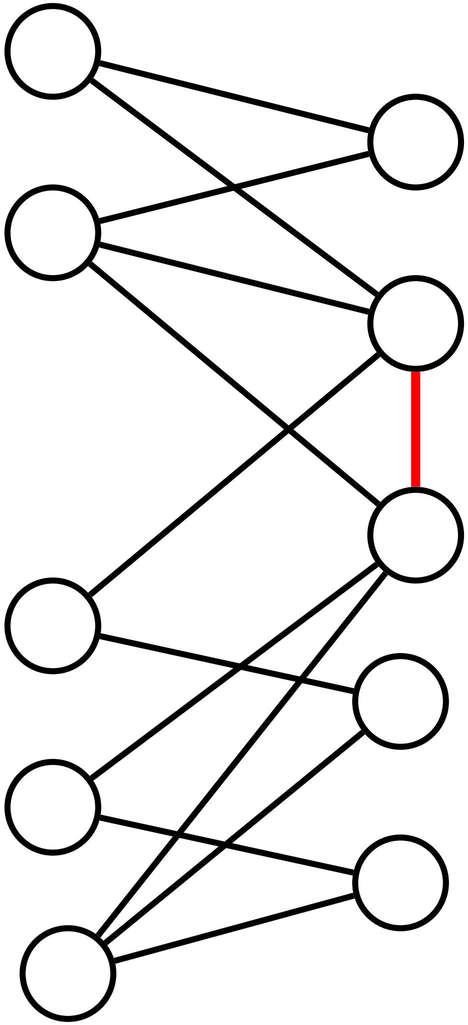
Search tree algorithm

- . . . or its other endpoint is in the optimal cover . . .



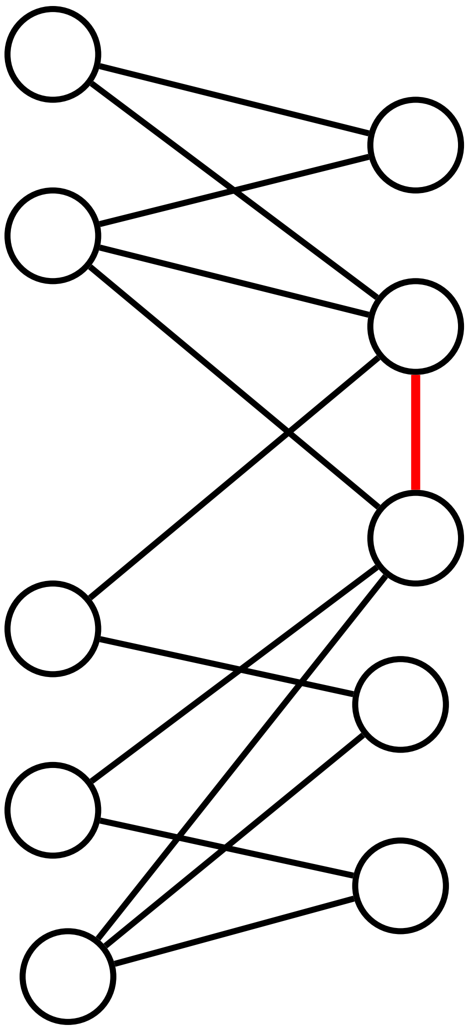
Search tree algorithm

- . . . or its other endpoint is in the optimal cover . . .
- So, we can remove it.



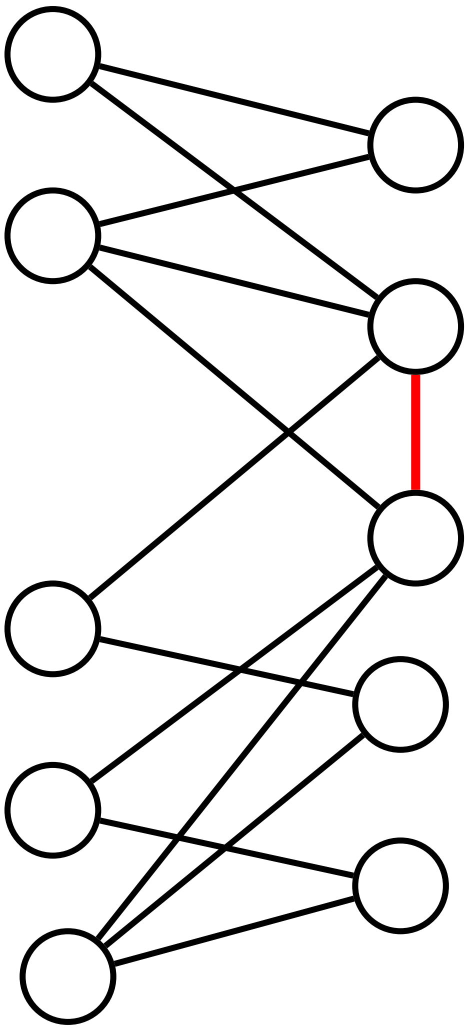
Search tree algorithm

- We have produced two instances, one equivalent to the original.
- Both are closer to being bipartite.



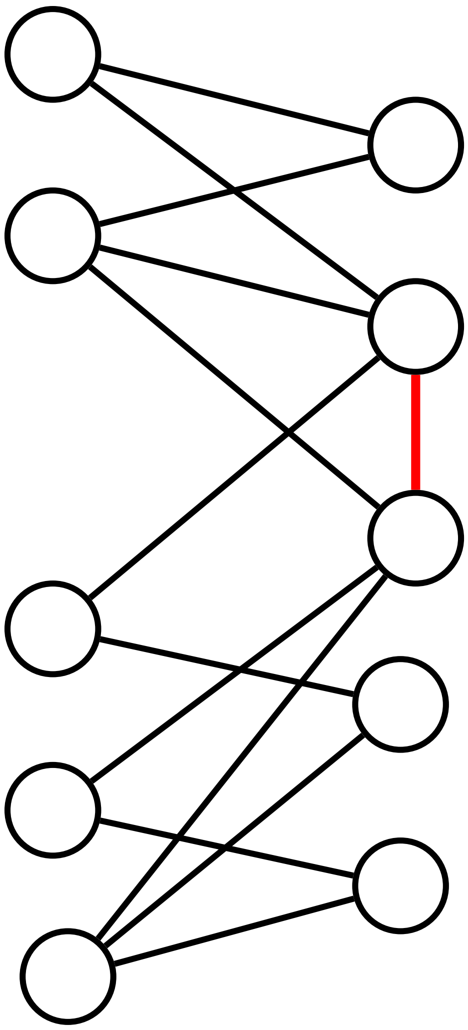
Search tree algorithm

- Continuing like this, we produce 2^k instances, where k is original distance from bipartite-ness.
- These are all bipartite. $\rightarrow$ Use poly-time algorithm to find the best.



Search tree algorithm

- This is known as a *bounded-depth search tree* algorithm. It's essentially a brute-force approach, confined to k.
- Total running time: $2^k n^c$.



So what?

- This is too easy! Hence, boring. . .
- This is just a cooked-up example. . .
- This isn't really new. . .

So what?

- This is too easy! Hence, boring. . .
  - This doesn't work for all problems! 3-coloring is NP-hard for $k = 3$ [Cai 2002]
- This is just a cooked-up example. . .
- This isn't really new. . .

So what?

- This is too easy! Hence, boring. . .
- This is just a cooked-up example. . .
  - True. But we can work this way with countless other problems/graph families. Some cases are bound to be interesting.
- This isn't really new. . .

So what?

- This is too easy! Hence, boring. . .
- This is just a cooked-up example. . .
- This isn't really new. . .
  - Novelty here is the pursuit of upper/lower bounds with respect to n and k.

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
n^c	MM, MST		

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d		
n^c	MM, MST		

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k)} n^c$	pS-VC, pBP-VC, pFVS-DS
n^c	MM, MST		

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k^3)}n^c$	pS-Treewidth
		$2^{O(k)}n^c$	pS-VC, pBP-VC, pFVS-DS
n^c	MM, MST		

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k^3)}n^c$	pS-Treewidth
		$2^{O(k)}n^c$	pS-VC, pBP-VC, pFVS-DS
		$2^{O(\sqrt{k})}n^c$	Planar pS-VC, Planar pS-DS, pS-FAST
n^c	MM, MST		

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k^3)} n^c$	pS-Treewidth
		$2^{O(k)} n^c$	pS-VC, pBP-VC, pFVS-DS
		$2^{O(\sqrt{k})} n^c$	Planar pS-VC, Planar pS-DS, pS-FAST
n^c	MM, MST		

NP-hardness

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k^3)} n^c$	pS-Treewidth
		$2^{O(k)} n^c$	pS-VC, pBP-VC, pFVS-DS
		$2^{O(\sqrt{k})} n^c$	Planar pS-VC, Planar pS-DS, pS-FAST
n^c	MM, MST		

NP-hardness

W-hardness

Parameterized Zoo

Classical		Parameterized	
Running time	Examples	Running time	Examples
$2^{O(n)}$	Clique, DS, TSP, VC		
$n^{poly \log n}$	VC-d	n^k	pS-DS, pS-Clique, pFVS-CapVC
		$2^{O(k^3)} n^c$	pS-Treewidth
		$2^{O(k)} n^c$	pS-VC, pBP-VC, pFVS-DS
		$2^{O(\sqrt{k})} n^c$	Planar pS-VC, Planar pS-DS, pS-FAST
n^c	MM, MST		

NP-hardness

W-hardness

ETH

Methodology

- First objective: prove that a problem is FPT ($f(k) \cdot n^c$)
  - Positive toolbox (algorithmic techniques)
  - Negative toolbox (W-hardness reductions)
    - Parameter-preserving reductions from known hard problems (Independent set, Dominating set . . .)
  - Second objective: get the best $f(k)$ ($2^{2^k} > 2^{k^2} > k^k > 3^k > 2^k$)
    - Positive toolbox (algorithmic techniques)
    - Negative toolbox (reductions from ETH)
      - The assumption that 3-SAT cannot be solved in $2^{o(n)}$.

What parameter to choose?

- We would like to define the parameter so that:
  - As many instances as possible have small k.
  - We can design an algorithm that works well for small k (FPT).
- These are conflicting goals! Picking a good parameter is hard work!
- One approach: “natural” parameterizations: k is the value of the objective function.

What parameter to choose?

- We would like to define the parameter so that:
  - As many instances as possible have small k.
  - We can design an algorithm that works well for small k (FPT).
- These are conflicting goals! Picking a good parameter is hard work!
- One approach: “natural” parameterizations: k is the value of the objective function.
- Here: Structural parameterizations: k is some measure of the complexity of the input graph/instance.
- Example: How about Vertex Cover in graphs with FVS of size k?

Graph Widths and Meta-Theorems

Structural Parameters

- Time to think a little harder about our choice of parameters.
- We will now investigate the algorithmic and graph-theoretic properties of various measures that quantify graph complexity.
  - ...an area known as the theory of graph "widths".

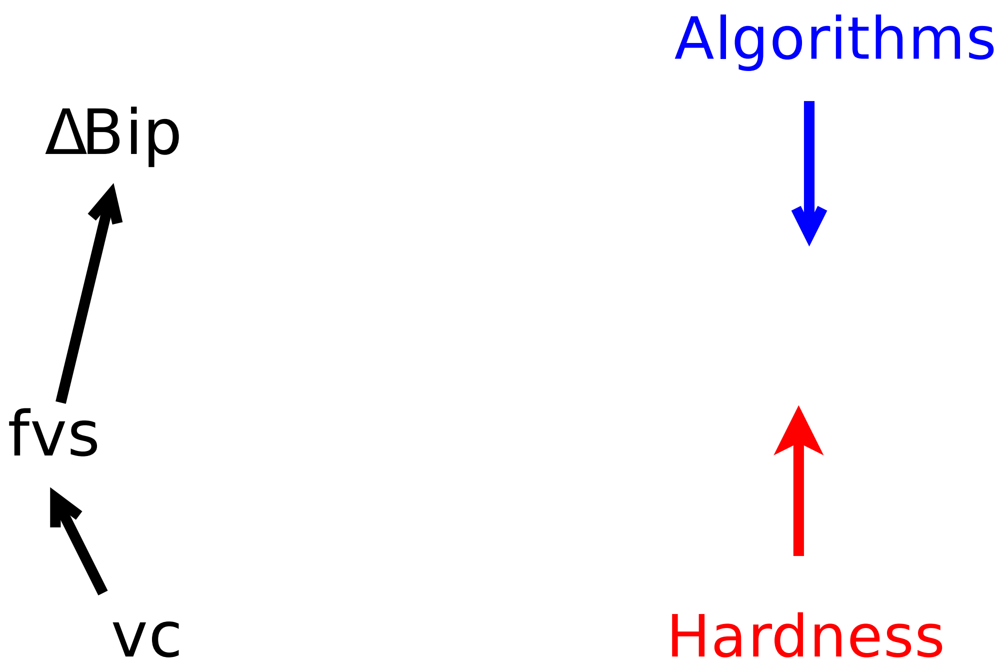
Graph widths

- The most popular structural parameters for graphs are the various graph "widths".
- Their king is treewidth.
  - Treewidth quantifies how "close" a graph is to being a tree.
  - Treewidth strikes a good balance between our two goals.
- A surprisingly robust notion, rediscovered independently several times
  - Arnborg and Proskurowski (partial k-trees), Robertson and Seymour (tree decompositions), Kirousis and Papadimitriou (node searching), . . .

Graph widths

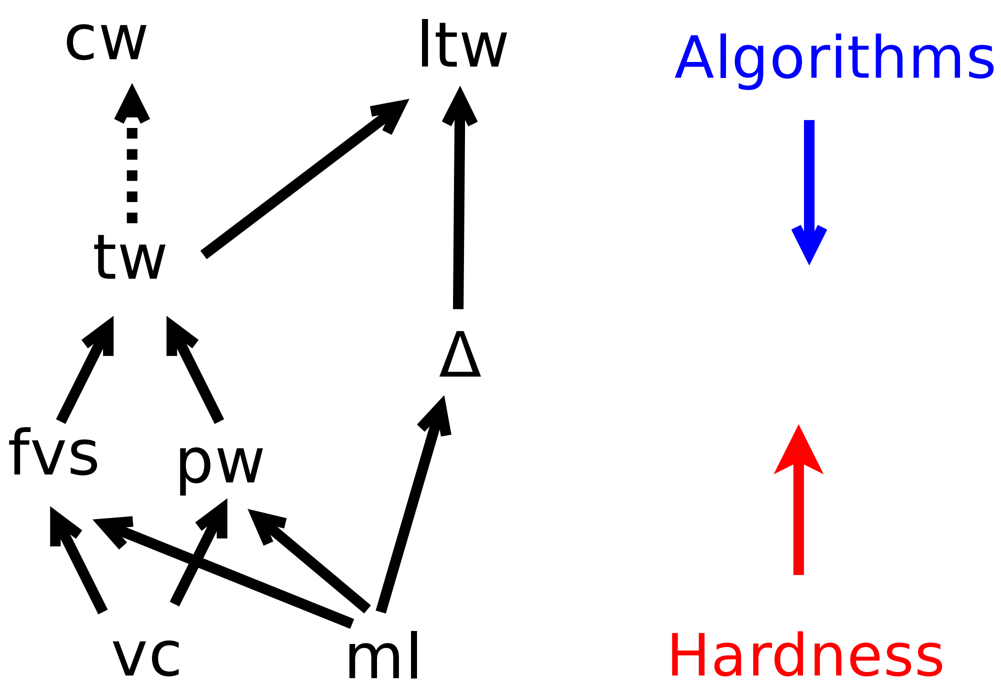
- The most popular structural parameters for graphs are the various graph "widths".
- Their king is treewidth.
  - Treewidth quantifies how "close" a graph is to being a tree.
  - Treewidth strikes a good balance between our two goals.
- What other "widths" are there? What are their properties? What are the relationships between them and with other graph invariants?

A complexity map



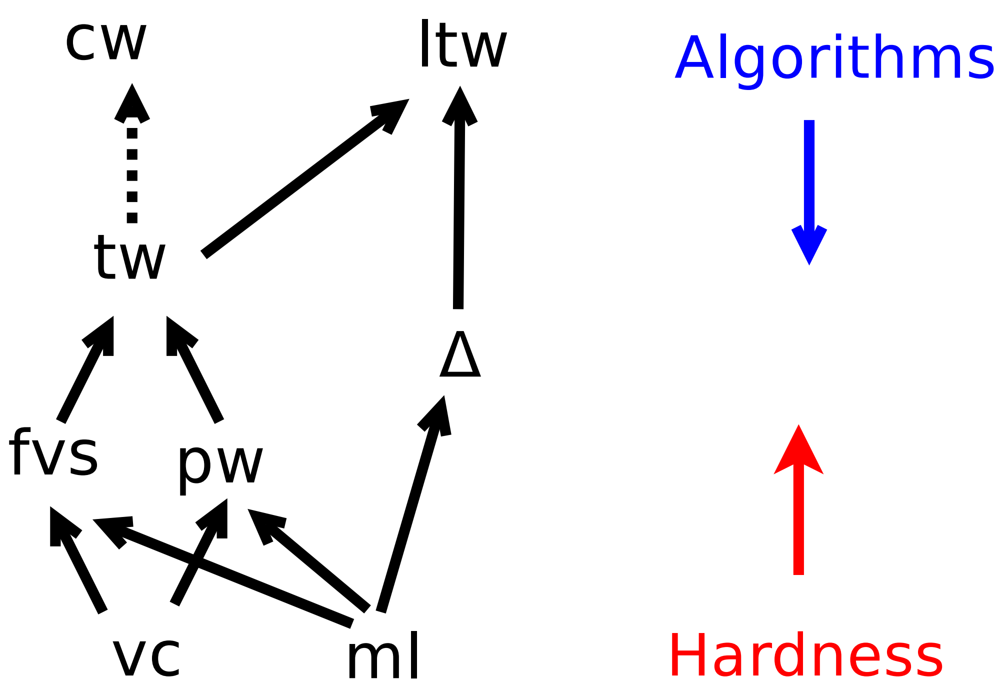
Recall two reasonable parameters. What is the trade-off between generality and algorithms?

A complexity map



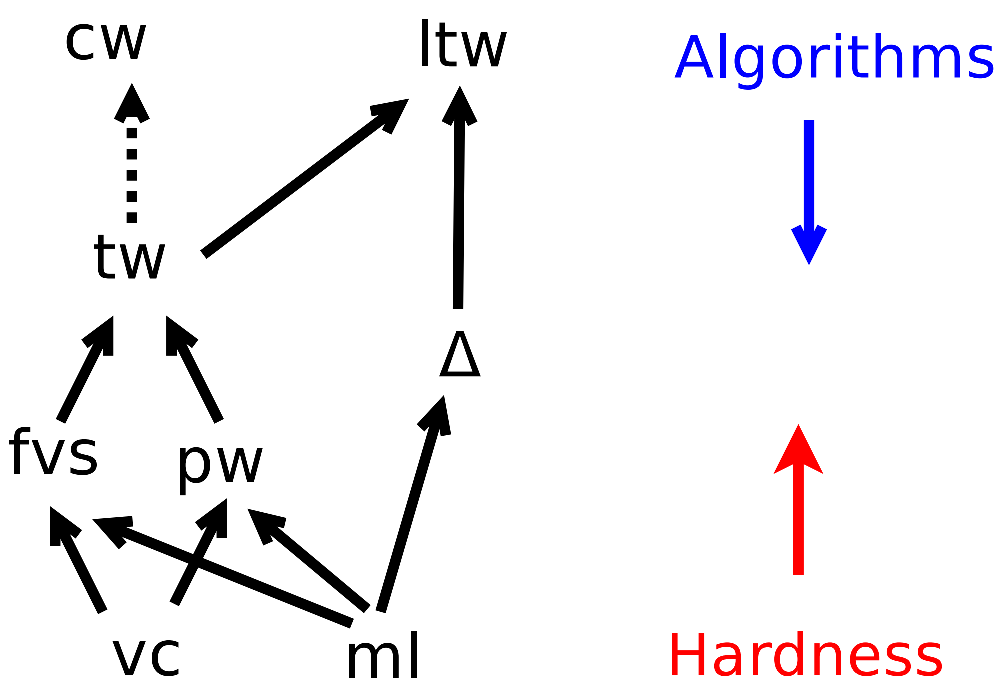
vc = Vertex Cover, ml = Max-Leaf, fvs = Feedback Vertex Set, pw = Pathwidth, tw = Treewidth, cw = Cliquewidth, ltw = Local Treewidth, Δ = Max Degree

A complexity map



Algorithmic implications: positive (FPT) results propagate downward, negative (hardness) results propagate upward

A complexity map



This map gives us a basic idea of how general each width is.

Algorithmic Meta-Theorems

- Algorithmic Theorems
  - Vertex Cover, Dominating Set, 3-Coloring are solvable in linear time on graphs of constant treewidth.

Algorithmic Meta-Theorems

- Algorithmic Meta-Theorems
  - All MSO-expressible problems are solvable in linear time on graphs of constant treewidth.
- Main uses: quick complexity classification tools, mapping the limits of applicability for specific techniques.
- Also: evaluating the algorithmic potency of a parameter.
- To prove such theorems we should be able to group families of problems together. Method here: expressibility in certain logics.

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Dominating Set of size 2

$$\exists x_1 \exists x_2 \forall y E(x_1, y) \vee E(x_2, y) \vee x_1 = y \vee x_2 = y$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Vertex Cover of size 2

$$\exists x_1 \exists x_2 \forall y \forall z E(y, z) \rightarrow (y = x_1 \vee y = x_2 \vee z = x_1 \vee z = x_2)$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Clique of size 3

$$\exists x_1 \exists x_2 \exists x_3 E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_1, x_3)$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Many standard (parameterized) problems can be expressed in FO logic. But some easy problems are inexpressible (e.g. connectivity).
Rule of thumb: FO = local properties

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO$_1$ logic: we can quantify over sets of vertices only
  - MSO$_2$ logic: we can quantify over sets of edges

Example: 2-coloring

$$\exists V_1 \exists V_2$$

$$(\forall x \forall y E(x,y) \rightarrow (x \in V_1 \leftrightarrow y \in V_2))$$

$$(\forall z (z \in V_1 \vee z \in V_2))$$

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges
- $\text{MSO}_2 \neq \text{MSO}_1$. Examples: Hamiltonicity, Edge dominating set

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges
- Optimization variants of MSO exist, questions of the form find min S s.t. $\phi(S)$ holds.

The model checking problem

Problem: **p-Model Checking**
Input: Graph G of width k and formula ϕ
Parameter: $|\phi| + k$
Question: $G \models \phi$?

- The unparameterized problem is PSPACE-hard, even for FO logic on trivial graphs.

The model checking problem

Problem: **p-Model Checking**
Input: Graph G of width k and formula ϕ
Parameter: $|\phi| + k$
Question: $G \models \phi$?

- If $|\phi|$ is a constant, problem is in XP for FO logic, NP-hard for MSO$_1$.
  - For FO logic, try all possibilities for each variable.
  - For MSO logic, 3-coloring is expressible with a constant-size formula.

The model checking problem

Problem: **p-Model Checking**
Input: Graph G of width k and formula ϕ
Parameter: $|\phi| + k$
Question: $G \models \phi$?

- Parameterized just by $|\phi|$, this problem is W-hard even for FO logic
    - The property "the graph has a clique of size t" can be encoded in an FO formula of size $O(t)$

The model checking problem

Problem: **p-Model Checking**
Input: Graph G of width k and formula ϕ
Parameter: $|\phi| + k$
Question: $G \models \phi$?

- We are interested in finding tractable, i.e. FPT, cases for the doubly parameterized case.

Courcelle's theorem

- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
  - Automata-theoretic proof, show that MSO graph properties have finite index.
  - Most celebrated result in this area. One of the reasons everyone loves treewidth.

Courcelle's theorem

- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
- More formally: There exists an algorithm which, given an MSO_2 formula ϕ and a graph G with n vertices and treewidth k decides if $G \models \phi$ in time $f(k, |\phi|) \cdot n$.

Courcelle's theorem

- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
- More formally: There exists an algorithm which, given an MSO_2 formula ϕ and a graph G with n vertices and treewidth k decides if $G \models \phi$ in time $f(k, |\phi|) \cdot n$.
- Can we do better?
  - Faster?
  - More graphs?
  - Wider logic?

Courcelle's theorem

- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
- More formally: There exists an algorithm which, given an MSO_2 formula ϕ and a graph G with n vertices and treewidth k decides if $G \models \phi$ in time $f(k, |\phi|) \cdot n$.
- Can we do better?
  - Faster?
    - Better than linear time is impossible! But f is a tower of exponentials with height proportional to $|\phi|$. Huge room for improvement?
  - More graphs?
  - Wider logic?

Courcelle's theorem

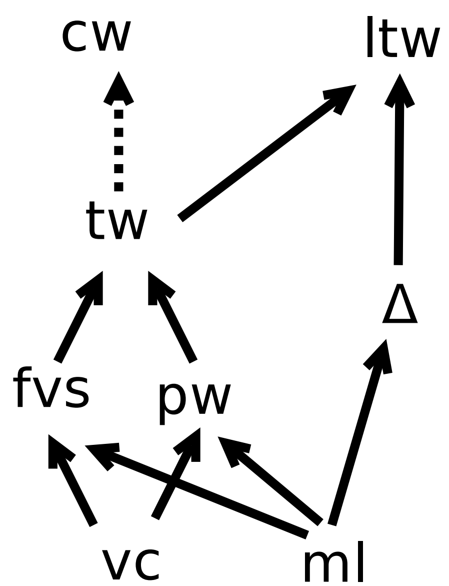
- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
- More formally: There exists an algorithm which, given an MSO_2 formula ϕ and a graph G with n vertices and treewidth k decides if $G \models \phi$ in time $f(k, |\phi|) \cdot n$.
- Can we do better?
  - Faster?
  - More graphs?
    - This has been extended to cliquewidth for MSO_1 logic [Courcelle, Makowsky, Rotics 2000]. It is impossible for MSO_2 [Fomin, Golovach, Lokshtanov, Saurabh 2009].
  - Wider logic?

Courcelle's theorem

- Every graph property expressible in MSO_2 logic is solvable in linear time on graphs of bounded treewidth.
- More formally: There exists an algorithm which, given an MSO_2 formula ϕ and a graph G with n vertices and treewidth k decides if $G \models \phi$ in time $f(k, |\phi|) \cdot n$.
- Can we do better?
  - Faster?
  - More graphs?
  - Wider logic?
    - ?

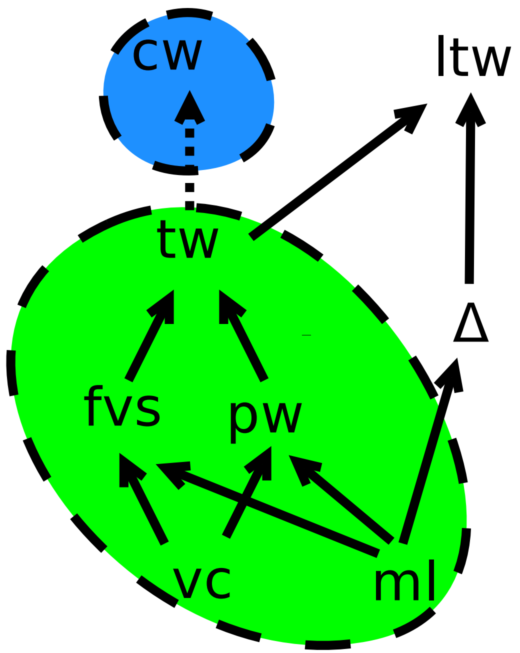
Vertex Cover and Max-Leaf

Graph classes



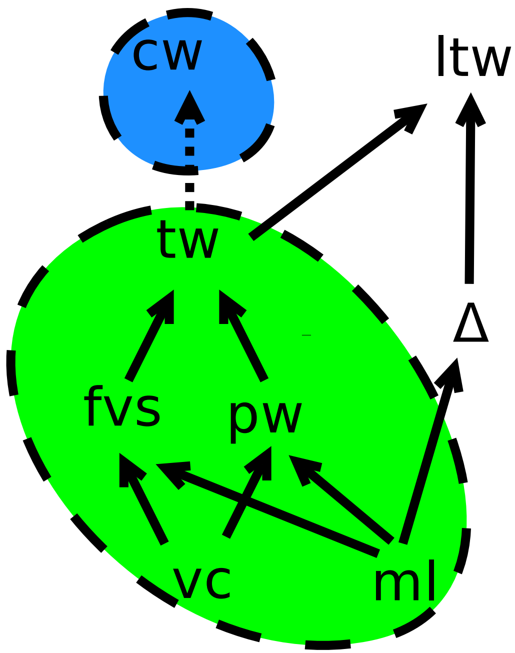
- FO logic is FPT for all, MSO_1 for the blue area, MSO_2 for the green area.
- FO logic is non-elementary for trees, triply exponential for binary trees. [Frick and Grohe 2004]

Graph classes



- FO logic is FPT for all, MSO_1 for the blue area, MSO_2 for the green area.
- FO logic is non-elementary for trees, triply exponential for binary trees. [Frick and Grohe 2004]

Graph classes



- FO logic is FPT for all, MSO_1 for the blue area, MSO_2 for the green area.
- FO logic is non-elementary for trees, triply exponential for binary trees. [Frick and Grohe 2004]

Our focus is on improving on the bottom.

Some newer meta-theorems

- FO logic for graphs of bounded vertex cover is singly exponential
- FO logic for graphs of bounded max-leaf number is singly exponential
- MSO logic for graphs of bounded vertex cover is doubly exponential
- Tight lower bounds (under the ETH) for vertex cover

([L. ESA 2010])

Vertex cover - FO

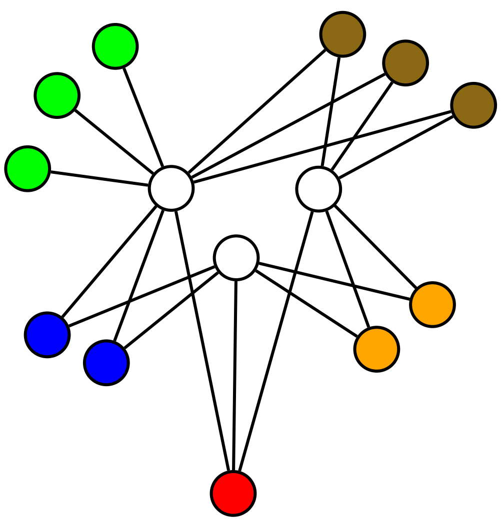
- Model checking FO logic on graphs of bounded vertex cover is singly exponential.

Vertex cover - FO

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Intuition:
  - Model checking FO logic on general graphs is in XP: each time we see a quantifier, we try all possible vertices.
  - The existence of a vertex cover of size k partitions the remainder of the graph into at most 2^k sets of vertices, depending on their neighbors in the vertex cover.
  - Crucial point: Trying all possible vertices in a set is wasteful. One representative suffices.

Vertex cover - FO

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Definition: u, v have the same type iff $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.
- Lemma: If $\phi(x)$ is a FO formula with a free variable and u, v have the same type then $G \models \phi(u)$ iff $G \models \phi(v)$.



Vertex cover - FO

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Algorithm: For each of the q quantified vertex variables in the formula try the following
  - Each of the vertices of the vertex cover (k choices)
  - Each of the previously selected vertices (q choices)
  - An arbitrary representative from each type (2^k choices)
- Total time: $O^*(k+q+2^k)^q = O^*(2^{kq+q\log q})$

Vertex cover - FO

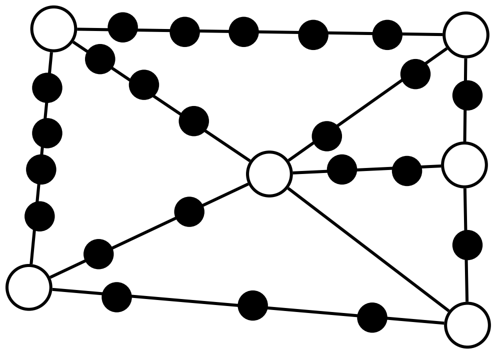
- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Algorithm: For each of the q quantified vertex variables in the formula try the following
  - Each of the vertices of the vertex cover (k choices)
  - Each of the previously selected vertices (q choices)
  - An arbitrary representative from each type (2^k choices)
- Total time: $O^*(k+q+2^k)^q = O^*(2^{kq+q\log q})$
- Recall: Courcelle's theorem gives a tower of exponentials here

Max-Leaf Number - FO

- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.

Max-Leaf Number - FO

- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.
- Again, small max-leaf number implies a special structure
  - Small degree and small pathwidth
  - [Kleitman and West 1991] A graph of max-leaf number k is a sub-division of a graph of at most $O(k)$ vertices.



Max-Leaf Number - FO

- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.
- Definition: a topo-edge is a vertex-maximal induced path
- The vast majority of vertices have degree 2 and belong in topo-edges

Max-Leaf Number - FO

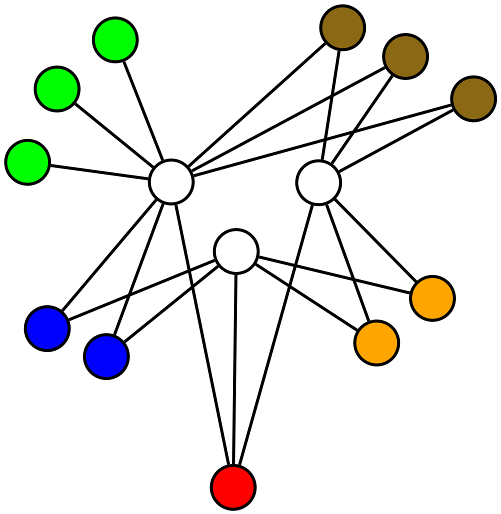
- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.
- Definition: a topo-edge is a vertex-maximal induced path
- The vast majority of vertices have degree 2 and belong in topo-edges
- Lemma: If a topo-edge has length at least 2^q it can be shortened without affecting the truth value of any FO sentence with at most q quantifiers.

Max-Leaf Number - FO

- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.
- Definition: a topo-edge is a vertex-maximal induced path
- The vast majority of vertices have degree 2 and belong in topo-edges
- Lemma: If a topo-edge has length at least 2^q it can be shortened without affecting the truth value of any FO sentence with at most q quantifiers.
- The graph can be reduced to size $O(k^2 2^q)$ so the trivial FO algorithm runs in $2^{O(q^2+q\log k)}$

Vertex Cover - MSO

- Again, using partition of vertices into types.
- To decide $\exists S \phi(S)$ we could try out all sets of vertices for S (2^n choices)
- But, the only thing that matters is how many vertices we pick from each type, not which.
- $\ldots n^{2^k}$ choices. Still too many. . .



Vertex Cover - MSO

- Again, using partition of vertices into types.
- Main idea: if there are more than 2^q vertices of a certain type, we can discard one.
- We end up with $2^k \cdot 2^q$ vertices. Deciding if an MSO sentence holds takes exponential time:
  - Total running time: $2^{2^{O(k+q)}}$

Vertex Cover - MSO

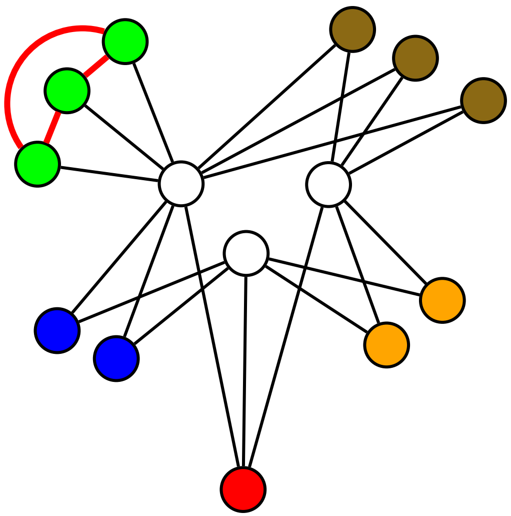
- Again, using partition of vertices into types.
- Main idea: if there are more than 2^q vertices of a certain type, we can discard one.
- We end up with $2^k \cdot 2^q$ vertices. Deciding if an MSO sentence holds takes exponential time:
  - Total running time: $2^{2^{O(k+q)}}$
- Lower bound argument shows that this cannot be improved to $2^{2^{o(k+q)}}$, assuming the ETH.

Generalizing

- The only property of graphs of small vertex cover that we use is that they can be partitioned into few equivalence types.

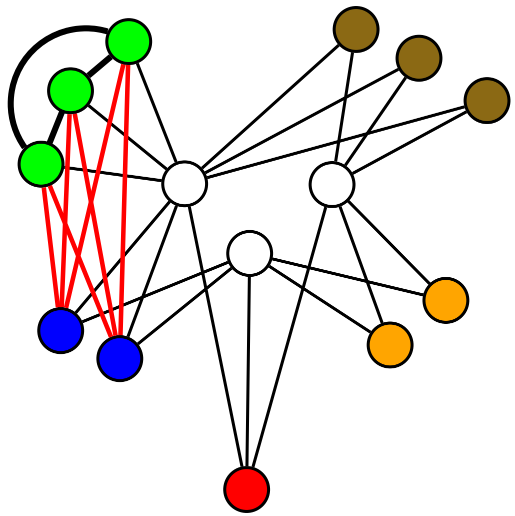
Generalizing

- The only property of graphs of small vertex cover that we use is that they can be partitioned into few equivalence types.
- Even if each type is not an independent set, its vertices are still equivalent.



Generalizing

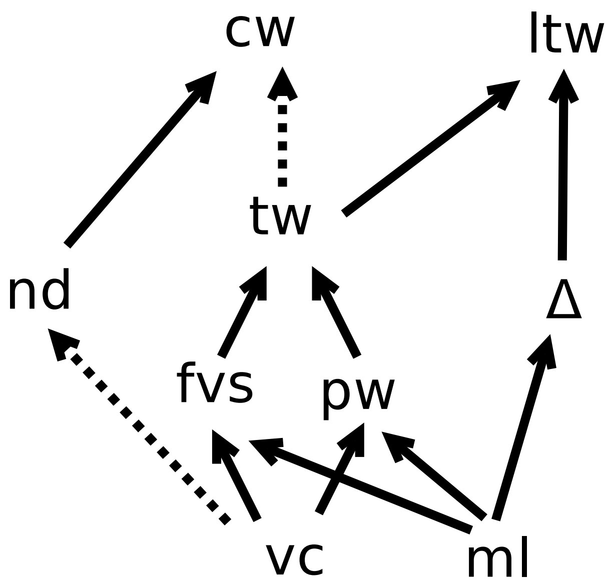
- The only property of graphs of small vertex cover that we use is that they can be partitioned into few equivalence types.
- Even if two types are connected, their vertices are still equivalent.



Generalizing

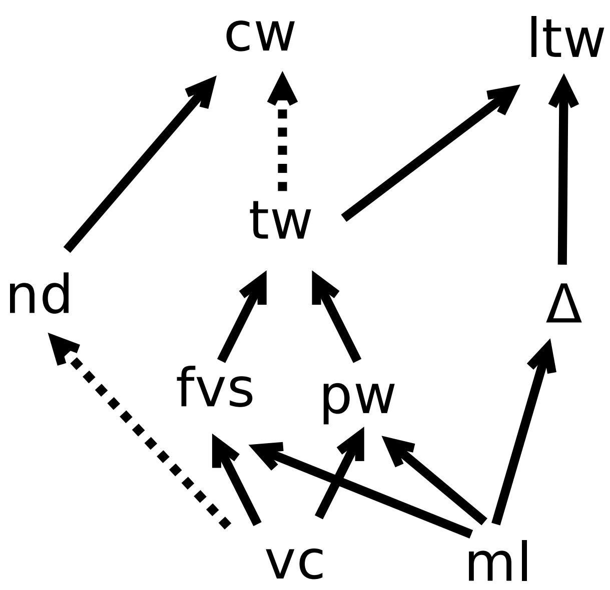
- The only property of graphs of small vertex cover that we use is that they can be partitioned into few equivalence types.
- Definition: The neighborhood diversity of a graph G is the number of type equivalence classes of its vertices.
- Each class may induce a clique or an independent set.
- Two classes are either disconnected or fully connected.
- $\mathrm{nd}(G)$ can be computed in polynomial time.

Neighborhood Diversity



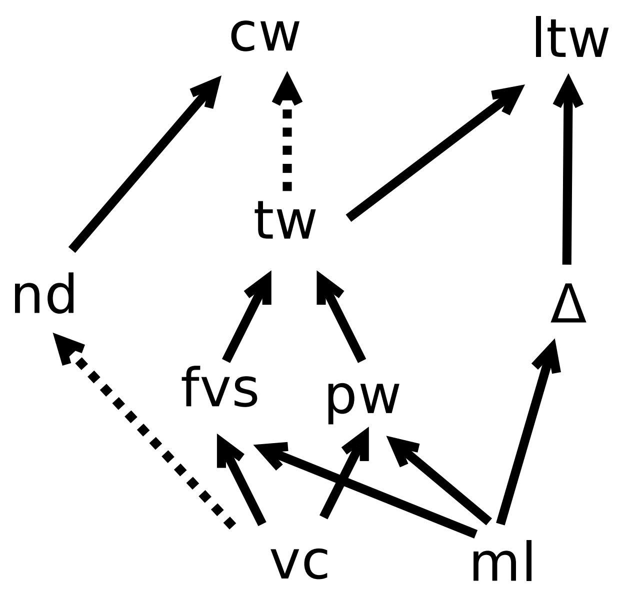
- All the meta-theorems for vertex cover naturally generalize to neighborhood diversity, with exponentially better running time.

Neighborhood Diversity



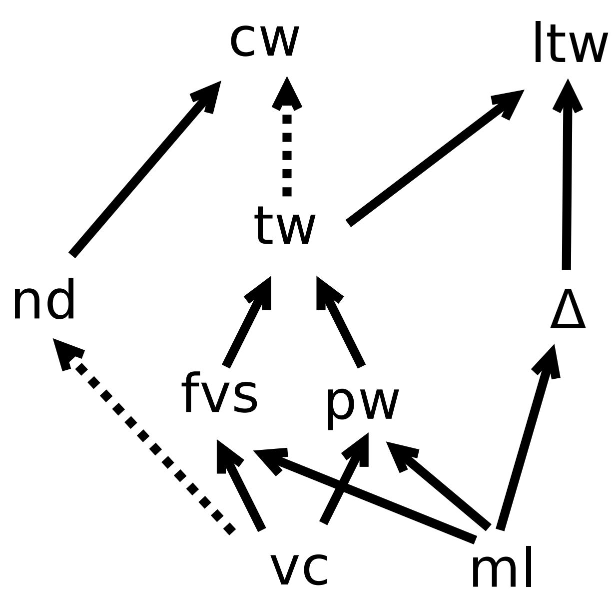
- nd is strictly more general than vertex cover, and incomparable to treewidth (think complete bipartite graphs).

Neighborhood Diversity



- It is a special case of clique-width. Several problems hard for clique-width are solvable for nd.

Neighborhood Diversity



- Is this a realistic parameter?
- Recently ([Ganian 2011]) a similar (but incomparable) generalization of vertex cover was suggested. Can these two be merged?

Conclusions

Open problems

- Structural parameterizations are a potentially large and still young research area.
- Need to explore more the properties of various widths.

Open problems

- Structural parameterizations are a potentially large and still young research area.
- Need to explore more the properties of various widths.
- Need to think harder about the way we define problem families.
  - What else is there besides FO and MSO logic?
  - Modal logic? [Pilipczuk 2011]

Open problems

- Structural parameterizations are a potentially large and still young research area.
- Need to explore more the properties of various widths.
- Need to think harder about the way we define problem families.
  - What else is there besides FO and MSO logic?
  - Modal logic? [Pilipczuk 2011]
- Concrete open problem:
  - MSO logic for max-leaf.
  - Interesting connections with (unary) regular language complexity.

The End

Thank you!

Questions?